CLAIMS

What is claimed is:

1. A system for evaluating and selecting programming code, comprising:

a first evaluator to measure a first characteristic of a plurality of input binaries and compute a plurality of first figures of merit for the plurality of input binaries, respectively; and

a binary selector to compare the plurality of first figure of merit and select one of the plurality of input binaries as having the highest or lowest overall figure of merit.

2. The system of claim 1 further comprising:

a second evaluator to measure a second characteristic of the plurality of input binaries and compute a plurality of second figures of merit for the plurality of input binaries, respectively, wherein

the binary selector is to compare the plurality of second figures of merit.

3. The system of claim 2 wherein the binary selector is to compute an overall figure of merit for each of the input binaries as a function of the input binary's first and second figures of merit.

4. The system of claim 2 further comprising:

a third evaluator to measure a third characteristic of the plurality of input binaries and compute a plurality of third figures of merit for the plurality of input binaries, respectively, wherein

the binary selector is to compare the plurality of third figures of merit.

5. The system of claim 2 wherein the first characteristic is performance, the greater the performance the smaller its associated figure of merit, and the second characteristic is compressed file size, the smaller the compressed file size the smaller its associated figure of merit.

6. The system of claim 2 wherein the binary selector is to compare the plurality of first and second figures of merit by computing a mathematical operation for each of the input binaries which includes the respective first and second figures of merit of that input binary.

7. The system of claim 6 wherein the first and second measured characteristics are selected from the group consisting of: performance, code size, power consumption, compressed file size, and memory footprint.

8. The system of claim 7 further comprising:

a code generator that includes a compiler and a linker to process an output of the compiler and produce the input binaries, wherein

the compiler exposes an optimization control to its user selected from the group consisting of: loop-unrolling; vectorization; and constant propagation.

9. The system of claim 8 wherein the code generator further comprises a binary rewriter to process an output of the linker and produce the input binaries, wherein the binary rewriter exposes an optimization control to its user selected from the group consisting of: constant propagation; code shrinking; and specialization.

10. The system of claim 8 further comprising:

a script processor to process an input script from the user, the script processor to read a plurality of optimization combinations from the input script and configure the code generator in accordance with the optimization combinations, wherein

the code generator is to produce the input binaries as configured by the optimization combinations, respectively.

11. The system of claim 7 further comprising:

a binary rewriter to produce the input binaries based on a source binary, wherein the binary rewriter is to expose optimization controls to its user.

12. The system of claim 11 further comprising;

a script processor to process an input script from the user, the script processor to read a plurality of optimization combinations from the input script and configure the binary rewriter in accordance with the optimization combinations, wherein

the binary rewriter is to produce the input binaries as configured by the optimization combinations, respectively.

13. A machine-implemented method for processing computer programming code, comprising:

a) producing a current version of a binary using a current optimization setting;

b) measuring a characteristic of the current version and computing a current figure of merit (FOM) associated with the current version;

c) comparing the current FOM with a previously computed FOM associated with a prior version of the binary; and

automatically repeating a)-c) for another optimization setting.

14. The method of claim 13 further comprising:

indicating to a user the version of the binary that has the highest or lowest FOM as determined from the comparisons.

15. The method of claim 14 further comprising:

ranking a plurality of versions of the binary in accordance with their respective FOMs as determined from the comparisons.

16. The method of claim 13 wherein the current and another optimization settings include optimization controls for compilation, linking, and binary rewriting, and wherein said producing comprises:

compiling source code and linking object files to produce an initial version of the binary, and rewriting the initial version into the current version, using the current optimization setting.

17. The method of claim 13 wherein the current and another optimization settings include optimization controls for compilation and linking, and wherein said producing comprises:

compiling source code and linking object files to produce the current version of the binary, using the current optimization setting.

18. The method of claim 13 wherein the current and another optimization settings include optimization controls for binary rewriting, and wherein said producing comprises:

rewriting an initial version of the binary into the current version, using the current optimization setting.

19. The method of claim 13 further comprising:

d) measuring another characteristic of the current version and computing another figure of merit (FOM) associated with said another characteristic and the current version; and

e) comparing said another FOM with a previously computed FOM that is associated with said another characteristic and with a prior version of the binary.

20. The method of claim 13 wherein the binary comprises a firmware driver, and the characteristic is compressed file size of the binary.

21. An article of manufacture comprising:

a machine-accessible medium containing instructions that, when executed, cause a machine to:

a) generate a binary under an optimization setting;

b) compute a cost as a function of a measured characteristic of the binary;

c) perform a) - b) a plurality of times each time with a different optimization setting but based on the same source program; and

d) compare the computed costs, to select the binary having the lowest overall cost.

22. The article of manufacture of claim 21 wherein the instructions cause the machine to perform a) - b) a plurality of times, by first compiling the source program and then rewriting the binary a plurality of times and then recompiling the source program and then rewriting the recompiled binary a plurality of times.

23. The article of manufacture of claim 21 further comprising instructions that cause the machine to perform b) by computing a further cost as a function of a measured, further characteristic of the binary generated in a).

24. The article of manufacture of claim 23 wherein the instructions cause the machine to compare the computed costs in d), by computing an overall cost for each generated binary, wherein the overall cost is a function of said cost and said further cost computed in b).